CONNECT 4

Documentation

Game 1: Connect 4

T1 Planning and Design based on Mathematical and Declarative Programming techniques:

In designing the Connect 4, mathematical structures and declarative programming techniques are pivotal in creating a functional, maintainable, and enjoyable gameplay experience. This game uses various software engineering principles and mathematical concepts to ensure that it operates smoothly.

Mathematical Structures:

Following mathematical structures were used:

- **Winning Conditions:** The Connect 4 game relies on mathematical checks to determine winning conditions. It systematically examines horizontal, vertical, and diagonal patterns on the game board, verifying whether a player has achieved four consecutive tokens to win the game.
- **Coordinate Parsing:** Mathematical parsing of user inputs translates alphanumeric coordinates (e.g., "A1", "B3") into corresponding row and column indices on the game board. This conversion enables precise placement of tokens and facilitates gameplay interactions.
- **Gravity Rules:** The game incorporates gravity rules to simulate token dropping mechanics. By analyzing the availability of spaces below the intended coordinate, the game ensures that tokens fall to the lowest possible position within a column, adhering to gravitational principles.

Declarative Programming Techniques:

Following declarative programming techniques were used:

- **CLI Presentation:** The game interface is designed using command-line output, presenting information and prompts in a structured format. User interactions, game status updates, and menu options are displayed sequentially, offering a clear and concise user experience.
- **Event-Driven Logic:** User input and game events trigger corresponding event-driven functions, such as player turns and menu selections. Each function

encapsulates a specific task or action, promoting a declarative programming style focused on defining what should happen in response to user actions.

- **State Management:** The game manages its state through iterative updates to the game board, player turns, and win conditions. By maintaining a consistent state representation and updating it incrementally, the game ensures coherence and integrity throughout the gameplay experience.

Software Engineering Principles:

In the making of this game, following software engineering principles were used:

- **Modularity and Encapsulation:** The Connect 4 game is structured with modular functions encapsulating specific tasks such as printing the game board, handling player turns, checking for a winner, and managing the game menu. This design promotes code reusability and facilitates easier maintenance.
- **Abstraction:** The game abstracts away implementation details from the user, providing a clean command-line interface (CLI) for interaction. Users are shielded from complexities such as coordinate parsing, gravity rules, and game logic, allowing for intuitive gameplay.
- **Scalability:** The game design supports scalability by offering multiple game modes, including a two-player mode and a mode against the computer (CPU). This flexibility accommodates varying player preferences and extends the game's longevity.

The planning and design of the Connect 4 game incorporate software engineering principles, mathematical concepts, and declarative programming techniques to create an interactive gaming experience. By modularizing the game, abstracting complex rules, using mathematical calculations, and designing a GUI, the game provides a great experience for players.

T2 Program Implementation based on Declarative Programming tools and techniques.

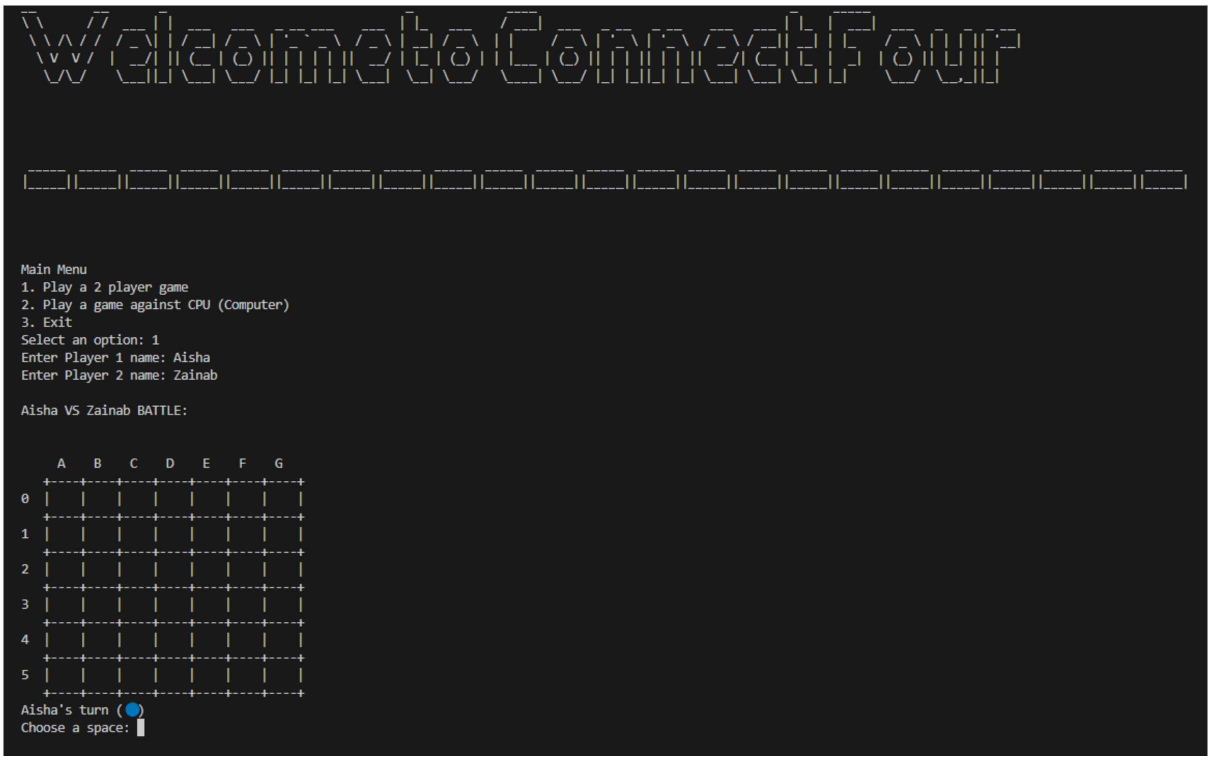
Source Code:

```python
'''

Connect 4 Game:
Programming Language: Python
Interface: CLI based
Student Name: Aisha Abdi
Student ID: 22160571

'''

import random
import art

# Print welcome message
art.tprint("Welcome to Connect Four")
art.tprint("-----------------------")

# Possible columns where players can place their tokens
possibleLetters = ["A", "B", "C", "D", "E", "F", "G"]
# Initialize the game board as a 6x7 grid
gameBoard = [["", "", "", "", "", "", ""],
             ["", "", "", "", "", "", ""],
             ["", "", "", "", "", "", ""],
             ["", "", "", "", "", "", ""],
             ["", "", "", "", "", "", ""],
             ["", "", "", "", "", "", ""]]

rows = 6
cols = 7

# Function to print the game board
def printGameBoard():
    print("\n     A    B    C    D    E    F    G  ", end="")
    for x in range(rows):
        print("\n   +----+----+----+----+----+----+----+")
        print(x, " |", end="")
        for y in range(cols):
            if gameBoard[x][y] == "●":
                print("", gameBoard[x][y], end=" |")
            elif gameBoard[x][y] == "●":
                print("", gameBoard[x][y], end=" |")
            else:
                print(" ", gameBoard[x][y], end="  |")
    print("\n   +----+----+----+----+----+----+----+")

# Function to modify the game board with the player's token
def modifyArray(spacePicked, turn):
    gameBoard[spacePicked[0]][spacePicked[1]] = turn

# Function to check for a winner
def checkForWinner(chip):
    # Check horizontal spaces
    for y in range(rows):
        for x in range(cols - 3):
            if gameBoard[x][y] == chip and gameBoard[x + 1][y] == chip and
gameBoard[x + 2][y] == chip and gameBoard[x + 3][y] == chip:
                print("\nGame over", chip, "wins! Thank you for playing :)")
                return True

    # Check vertical spaces
    for x in range(rows):
        for y in range(cols - 3):
            if gameBoard[x][y] == chip and gameBoard[x][y + 1] == chip and
gameBoard[x][y + 2] == chip and gameBoard[x][y + 3] == chip:
                print("\nGame over", chip, "wins! Thank you for playing :)")
                return True

    # Check upper right to bottom left diagonal spaces
    for x in range(rows - 3):
        for y in range(3, cols):
            if gameBoard[x][y] == chip and gameBoard[x + 1][y - 1] == chip and
gameBoard[x + 2][y - 2] == chip and gameBoard[x + 3][y - 3] == chip:
                print("\nGame over", chip, "wins! Thank you for playing :)")
                return True

    # Check upper left to bottom right diagonal spaces
    for x in range(rows - 3):
        for y in range(cols - 3):
            if gameBoard[x][y] == chip and gameBoard[x + 1][y + 1] == chip and
gameBoard[x + 2][y + 2] == chip and gameBoard[x + 3][y + 3] == chip:
                print("\nGame over", chip, "wins! Thank you for playing :)")
                return True
    return False

# Function to convert user input (e.g., "A1") to board coordinates
def coordinateParser(inputString):
    if len(inputString) != 2:
        raise ValueError("Invalid input format. Input should be a letter
followed by a number (e.g., A1).")
```

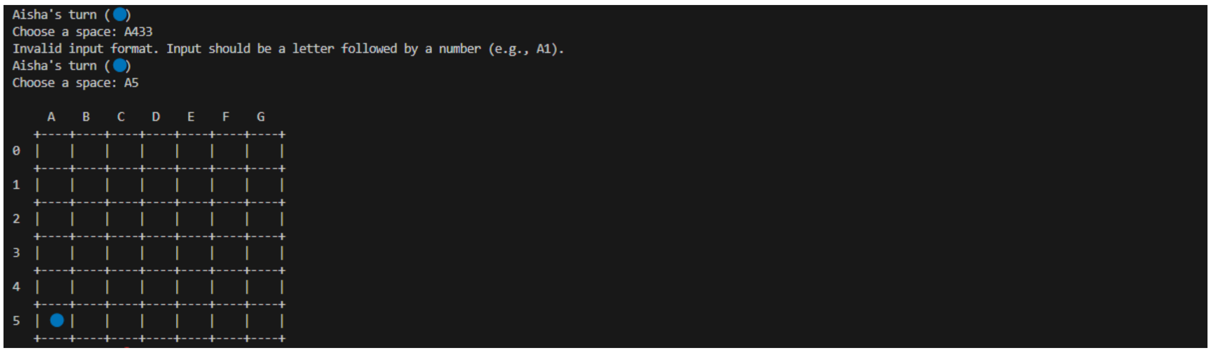
```python
    column = inputString[0].upper()
    if column not in possibleLetters:
        raise ValueError("Invalid column letter.")
    row = int(inputString[1])
    if row < 0 or row >= rows:
        raise ValueError("Invalid row number.")
    return [row, possibleLetters.index(column)]

# Function to check if a space on the board is available
def isSpaceAvailable(intendedCoordinate):
    if gameBoard[intendedCoordinate[0]][intendedCoordinate[1]] == '●':
        return False
    elif gameBoard[intendedCoordinate[0]][intendedCoordinate[1]] == '●':
        return False
    else:
        return True

# Function to check if a space is valid according to gravity rules
def gravityChecker(intendedCoordinate):
    # Calculate space below
    spaceBelow = [intendedCoordinate[0] + 1, intendedCoordinate[1]]
    # Check if the coordinate is at ground level
    if spaceBelow[0] == 6:
        return True
    # Check if there's a token below
    if isSpaceAvailable(spaceBelow) == False:
        return True
    return False

# Function to handle a player's turn
def playerTurn(playerName, playerChip):
    while True:
        print(f"{playerName}'s turn ({playerChip})")
        spacePicked = input("Choose a space: ")
        try:
            coordinate = coordinateParser(spacePicked)
            # Check if the space is available and valid according to gravity
rules
            if isSpaceAvailable(coordinate) and gravityChecker(coordinate):
                modifyArray(coordinate, playerChip)
                break
            else:
                print("Not a valid coordinate")
        except ValueError as e:
            print(e)

# Function to handle the computer's turn
def computerTurn():
    print("Computer's turn (●)")
    while True:
        cpuChoice = [random.choice(possibleLetters), random.randint(0, 5)]
        cpuCoordinate = coordinateParser(cpuChoice[0] + str(cpuChoice[1]))
        if isSpaceAvailable(cpuCoordinate) and gravityChecker(cpuCoordinate):
            modifyArray(cpuCoordinate, '●')
            break

# Function to start a two-player game
def twoPlayerGame():
    player1Name = input("Enter Player 1 name: ")
    player2Name = input("Enter Player 2 name: ")
    print(f"\n{player1Name} VS {player2Name} BATTLE:\n")
    leaveLoop = False
    turnCounter = 0
    while not leaveLoop:
        printGameBoard()
        if turnCounter % 2 == 0:
            playerTurn(player1Name, '●')
            winner = checkForWinner('●')
        else:
            playerTurn(player2Name, '●')
            winner = checkForWinner('●')
        turnCounter += 1
        if winner:
            printGameBoard()
            break

# Function to start a game against the computer
def playAgainstCPU():
    playerName = input("Enter Player name: ")
    print(f"\n{playerName} VS Computer BATTLE:\n")
    leaveLoop = False
    turnCounter = 0
    while not leaveLoop:
        printGameBoard()
        if turnCounter % 2 == 0:
            playerTurn(playerName, '●')
            winner = checkForWinner('●')
        else:
            computerTurn()
            winner = checkForWinner('●')
        turnCounter += 1
        if winner:
            printGameBoard()
            break

def mainMenu():
```

```python
        print("\nMain Menu")
        print("1. Play a 2 player game")
        print("2. Play a game against CPU (Computer)")
        print("3. Exit")
        choice = input("Select an option: ")
        if choice == "1":
            twoPlayerGame()
        elif choice == "2":
            playAgainstCPU()
        elif choice == "3":
            print("Thank you for playing! Goodbye.")
            exit(0)
        else:
            print("Invalid choice. Please select again.")

mainMenu()
```

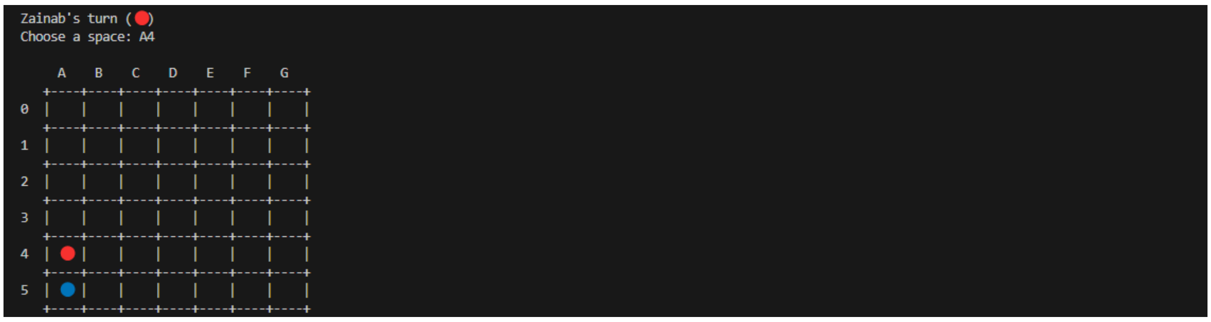
Output Screenshots:



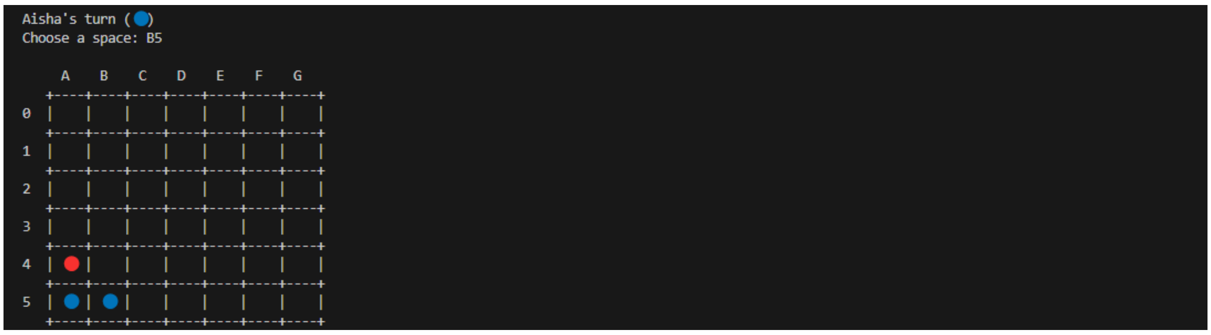
```
__        __   _                            _           ____                            _     _____
\ \      / /__| | ___ ___  _ __ ___   ___  | |_ ___    / ___|___  _ __  _ __   ___  ___| |_  |  ___|__  _   _ _ __
 \ \ /\ / / _ \ |/ __/ _ \| '_ ` _ \ / _ \ | __/ _ \  | |   / _ \| '_ \| '_ \ / _ \/ __| __| | |_ / _ \| | | | '__|
  \ V  V /  __/ | (_| (_) | | | | | |  __/ | || (_) | | |__| (_) | | | | | | |  __/ (__| |_  |  _| (_) | |_| | |
   \_/\_/ \___|_|\___\___/|_| |_| |_|\___|  \__\___/   \____\___/|_| |_|_| |_|\___|\___|\__| |_|  \___/ \__,_|_|



 _____  _____  _____  _____  _____  _____  _____  _____  _____  _____  _____  _____  _____  _____  _____  _____  _____  _____  _____  _____  _____  _____  _____
|_____||_____||_____||_____||_____||_____||_____||_____||_____||_____||_____||_____||_____||_____||_____||_____||_____||_____||_____||_____||_____||_____||_____|



Main Menu
1. Play a 2 player game
2. Play a game against CPU (Computer)
3. Exit
Select an option: 1
Enter Player 1 name: Aisha
Enter Player 2 name: Zainab

Aisha VS Zainab BATTLE:


     A    B    C    D    E    F    G
  +----+----+----+----+----+----+----+
0 |    |    |    |    |    |    |    |
  +----+----+----+----+----+----+----+
1 |    |    |    |    |    |    |    |
  +----+----+----+----+----+----+----+
2 |    |    |    |    |    |    |    |
  +----+----+----+----+----+----+----+
3 |    |    |    |    |    |    |    |
  +----+----+----+----+----+----+----+
4 |    |    |    |    |    |    |    |
  +----+----+----+----+----+----+----+
5 |    |    |    |    |    |    |    |
  +----+----+----+----+----+----+----+
Aisha's turn (🔵)
Choose a space:
```



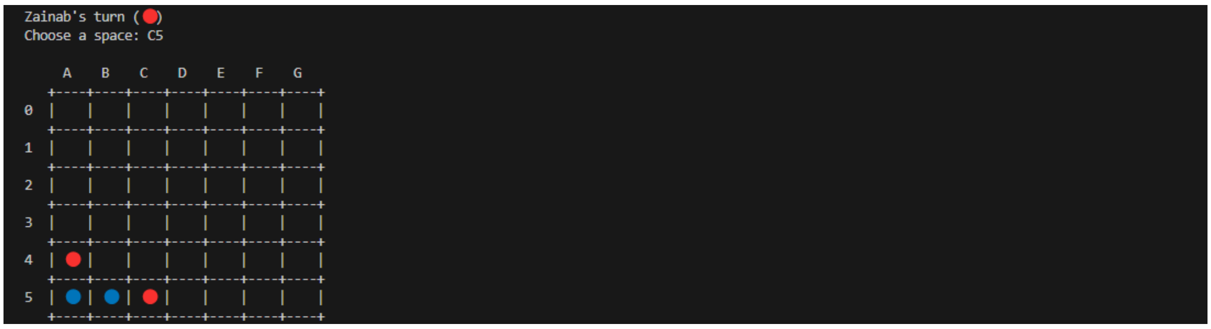
```
Aisha's turn (🔵)
Choose a space: A433
Invalid input format. Input should be a letter followed by a number (e.g., A1).
Aisha's turn (🔵)
Choose a space: A5

     A    B    C    D    E    F    G
  +----+----+----+----+----+----+----+
0 |    |    |    |    |    |    |    |
  +----+----+----+----+----+----+----+
1 |    |    |    |    |    |    |    |
  +----+----+----+----+----+----+----+
2 |    |    |    |    |    |    |    |
  +----+----+----+----+----+----+----+
3 |    |    |    |    |    |    |    |
  +----+----+----+----+----+----+----+
4 |    |    |    |    |    |    |    |
  +----+----+----+----+----+----+----+
5 | 🔵 |    |    |    |    |    |    |
  +----+----+----+----+----+----+----+
```



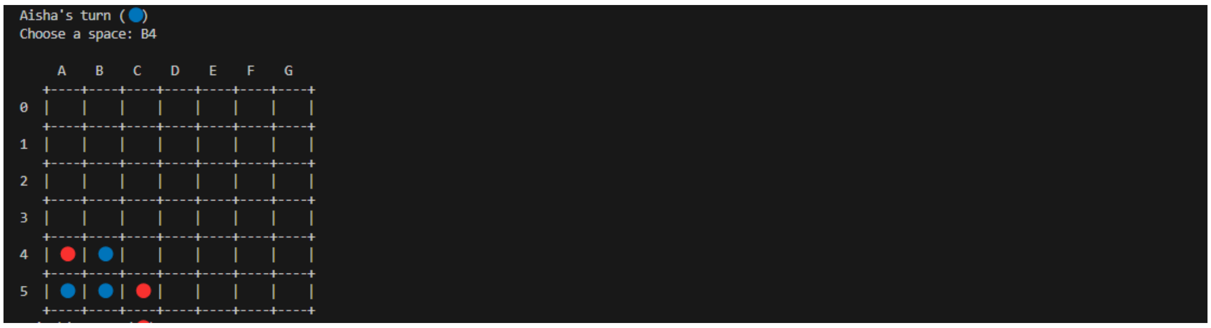
```
Zainab's turn (🔴)
Choose a space: A4

     A    B    C    D    E    F    G
  +----+----+----+----+----+----+----+
0 |    |    |    |    |    |    |    |
  +----+----+----+----+----+----+----+
1 |    |    |    |    |    |    |    |
  +----+----+----+----+----+----+----+
2 |    |    |    |    |    |    |    |
  +----+----+----+----+----+----+----+
3 |    |    |    |    |    |    |    |
  +----+----+----+----+----+----+----+
4 | 🔴 |    |    |    |    |    |    |
  +----+----+----+----+----+----+----+
5 | 🔵 |    |    |    |    |    |    |
  +----+----+----+----+----+----+----+
```



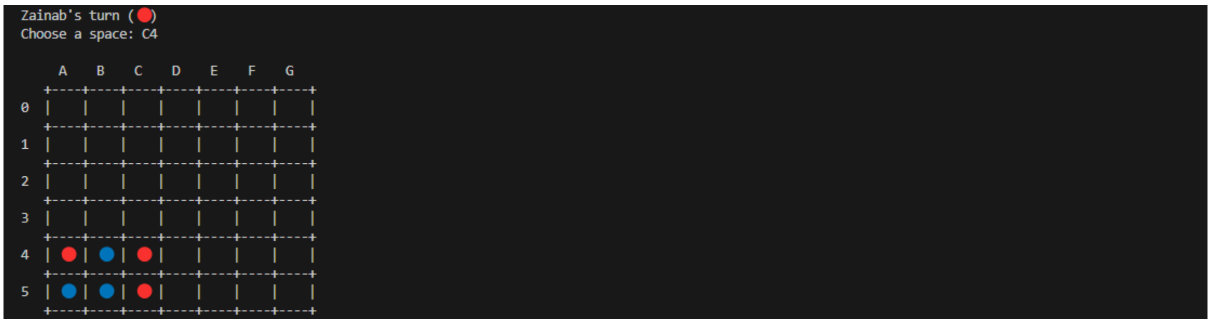
```
Aisha's turn (🔵)
Choose a space: B5

     A    B    C    D    E    F    G
  +----+----+----+----+----+----+----+
0 |    |    |    |    |    |    |    |
  +----+----+----+----+----+----+----+
1 |    |    |    |    |    |    |    |
  +----+----+----+----+----+----+----+
2 |    |    |    |    |    |    |    |
  +----+----+----+----+----+----+----+
3 |    |    |    |    |    |    |    |
  +----+----+----+----+----+----+----+
4 | 🔴 |    |    |    |    |    |    |
  +----+----+----+----+----+----+----+
5 | 🔵 | 🔵 |    |    |    |    |    |
  +----+----+----+----+----+----+----+
```



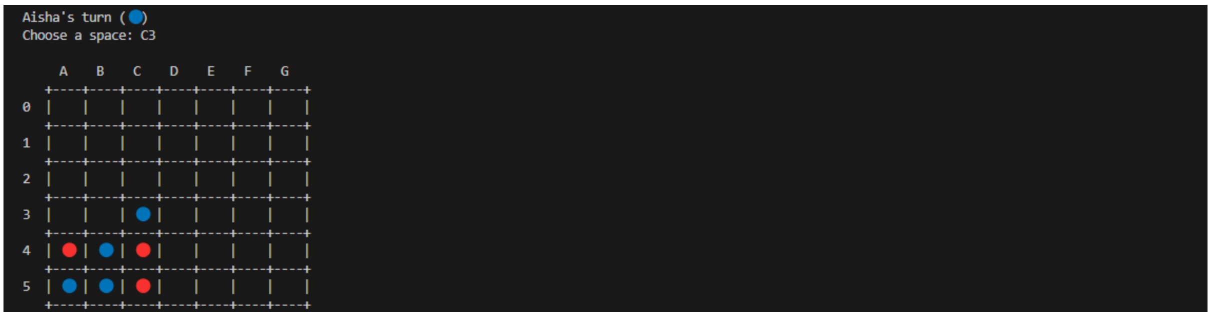
```
Zainab's turn (🔴)
Choose a space: C5

     A    B    C    D    E    F    G
   +----+----+----+----+----+----+----+
0  |    |    |    |    |    |    |    |
   +----+----+----+----+----+----+----+
1  |    |    |    |    |    |    |    |
   +----+----+----+----+----+----+----+
2  |    |    |    |    |    |    |    |
   +----+----+----+----+----+----+----+
3  |    |    |    |    |    |    |    |
   +----+----+----+----+----+----+----+
4  | 🔴 |    |    |    |    |    |    |
   +----+----+----+----+----+----+----+
5  | 🔵 | 🔵 | 🔴 |    |    |    |    |
   +----+----+----+----+----+----+----+
```



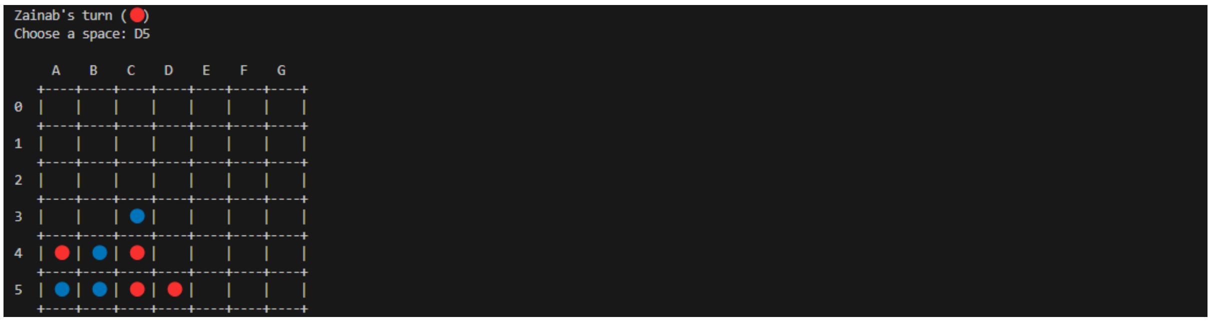
```
Aisha's turn (🔵)
Choose a space: B4

     A    B    C    D    E    F    G
   +----+----+----+----+----+----+----+
0  |    |    |    |    |    |    |    |
   +----+----+----+----+----+----+----+
1  |    |    |    |    |    |    |    |
   +----+----+----+----+----+----+----+
2  |    |    |    |    |    |    |    |
   +----+----+----+----+----+----+----+
3  |    |    |    |    |    |    |    |
   +----+----+----+----+----+----+----+
4  | 🔴 | 🔵 |    |    |    |    |    |
   +----+----+----+----+----+----+----+
5  | 🔵 | 🔵 | 🔴 |    |    |    |    |
   +----+----+----+----+----+----+----+
```



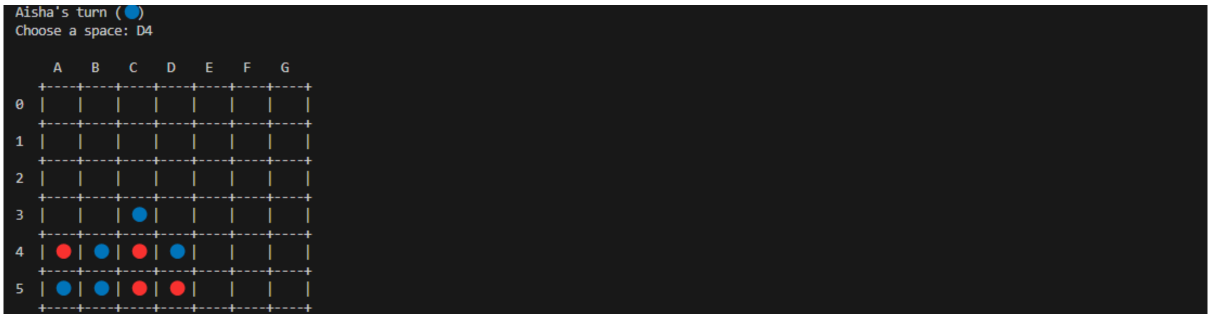
```
Zainab's turn (🔴)
Choose a space: C4

     A    B    C    D    E    F    G
   +----+----+----+----+----+----+----+
0  |    |    |    |    |    |    |    |
   +----+----+----+----+----+----+----+
1  |    |    |    |    |    |    |    |
   +----+----+----+----+----+----+----+
2  |    |    |    |    |    |    |    |
   +----+----+----+----+----+----+----+
3  |    |    |    |    |    |    |    |
   +----+----+----+----+----+----+----+
4  | 🔴 | 🔵 | 🔴 |    |    |    |    |
   +----+----+----+----+----+----+----+
5  | 🔵 | 🔵 | 🔴 |    |    |    |    |
   +----+----+----+----+----+----+----+
```



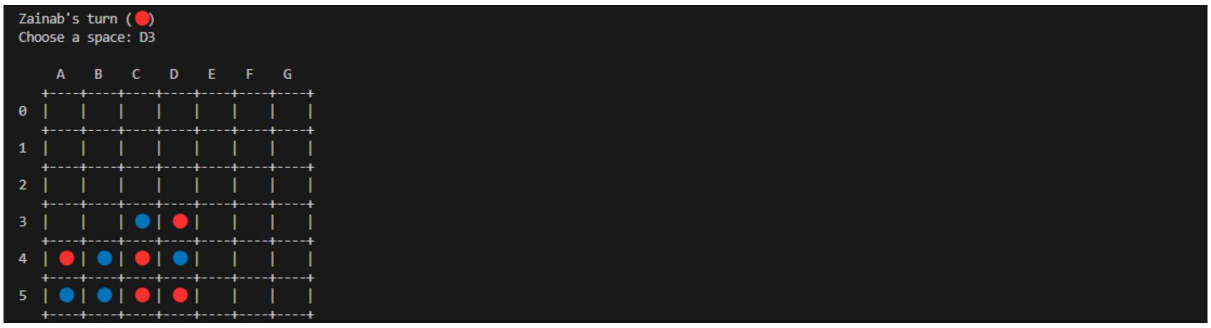
```
Aisha's turn (🔵)
Choose a space: C3

     A    B    C    D    E    F    G
   +----+----+----+----+----+----+----+
0  |    |    |    |    |    |    |    |
   +----+----+----+----+----+----+----+
1  |    |    |    |    |    |    |    |
   +----+----+----+----+----+----+----+
2  |    |    |    |    |    |    |    |
   +----+----+----+----+----+----+----+
3  |    |    | 🔵 |    |    |    |    |
   +----+----+----+----+----+----+----+
4  | 🔴 | 🔵 | 🔴 |    |    |    |    |
   +----+----+----+----+----+----+----+
5  | 🔵 | 🔵 | 🔴 |    |    |    |    |
   +----+----+----+----+----+----+----+
```



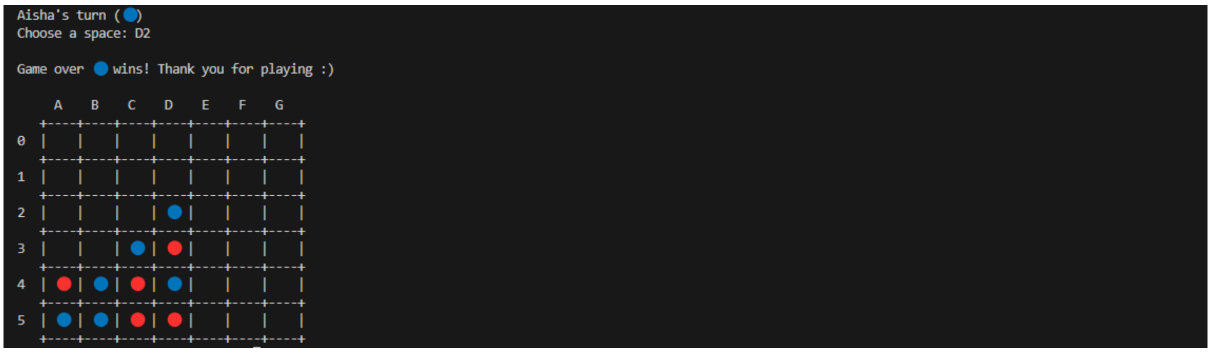
```
Zainab's turn (🔴)
Choose a space: D5

     A    B    C    D    E    F    G
   +----+----+----+----+----+----+----+
0  |    |    |    |    |    |    |    |
   +----+----+----+----+----+----+----+
1  |    |    |    |    |    |    |    |
   +----+----+----+----+----+----+----+
2  |    |    |    |    |    |    |    |
   +----+----+----+----+----+----+----+
3  |    |    | 🔵 |    |    |    |    |
   +----+----+----+----+----+----+----+
4  | 🔴 | 🔵 | 🔴 |    |    |    |    |
   +----+----+----+----+----+----+----+
5  | 🔵 | 🔵 | 🔴 | 🔴 |    |    |    |
   +----+----+----+----+----+----+----+
```



```
Aisha's turn (🔵)
Choose a space: D4

     A    B    C    D    E    F    G
  +----+----+----+----+----+----+----+
0 |    |    |    |    |    |    |    |
  +----+----+----+----+----+----+----+
1 |    |    |    |    |    |    |    |
  +----+----+----+----+----+----+----+
2 |    |    |    |    |    |    |    |
  +----+----+----+----+----+----+----+
3 |    |    | 🔵 |    |    |    |    |
  +----+----+----+----+----+----+----+
4 | 🔴 | 🔵 | 🔴 | 🔵 |    |    |    |
  +----+----+----+----+----+----+----+
5 | 🔵 | 🔵 | 🔴 | 🔴 |    |    |    |
  +----+----+----+----+----+----+----+
```



```
Zainab's turn (🔴)
Choose a space: D3

     A    B    C    D    E    F    G
  +----+----+----+----+----+----+----+
0 |    |    |    |    |    |    |    |
  +----+----+----+----+----+----+----+
1 |    |    |    |    |    |    |    |
  +----+----+----+----+----+----+----+
2 |    |    |    |    |    |    |    |
  +----+----+----+----+----+----+----+
3 |    |    | 🔵 | 🔴 |    |    |    |
  +----+----+----+----+----+----+----+
4 | 🔴 | 🔵 | 🔴 | 🔵 |    |    |    |
  +----+----+----+----+----+----+----+
5 | 🔵 | 🔵 | 🔴 | 🔴 |    |    |    |
  +----+----+----+----+----+----+----+
```



```
Aisha's turn (🔵)
Choose a space: D2

Game over 🔵 wins! Thank you for playing :)

     A    B    C    D    E    F    G
  +----+----+----+----+----+----+----+
0 |    |    |    |    |    |    |    |
  +----+----+----+----+----+----+----+
1 |    |    |    |    |    |    |    |
  +----+----+----+----+----+----+----+
2 |    |    |    | 🔵 |    |    |    |
  +----+----+----+----+----+----+----+
3 |    |    | 🔵 | 🔴 |    |    |    |
  +----+----+----+----+----+----+----+
4 | 🔴 | 🔵 | 🔴 | 🔵 |    |    |    |
  +----+----+----+----+----+----+----+
5 | 🔵 | 🔵 | 🔴 | 🔴 |    |    |    |
  +----+----+----+----+----+----+----+
```

Declarative Implementation of Connect Four in Python:

In a declarative programming style, The focus is on describing what the program should accomplish rather than detailing the control flow.

Data Structures and Type Definitions:

To effectively manage game data, the following data structures will be used:

- **Game Board:** Represented as a list of lists, where each sublist corresponds to a row on the board. Each element can be either empty (None) or contain a player's chip ('🔵' or '🔴').
- **Player:** Represented as a dictionary with keys for the player's name and chip symbol.
- **Coordinates:** Represented as tuples (row, column).

Behaviour Step Implementations:

- **Printing the Game Board:** A function to print the current state of the game board using list comprehensions to construct each row.

- **Coordinate Parsing:** Convert user input (e.g., "A1") to board coordinates. We handle invalid input using exceptions.
- **Checking for a Winner:** Functions to check for horizontal, vertical, and diagonal wins. Using list comprehensions to check sequences of four chips.
- **Placing a Chip:** A function to place a chip on the board following the gravity rule, ensuring the chip falls to the lowest available row in the selected column.
- **Player Turn Handling:** Functions to handle player input, validate moves, and update the board state.
- **Main Game Logic:** The main function switches turns between players and checking for a winner after each move in a loop.

Performance and Optimizations

- **Data Structures:** Using lists for the board ensures O(1) access time for reading and writing cells, which is efficient for our purposes.
- **Parsing and Validation:** Input parsing and move validation are designed to be efficient and straightforward, using exceptions for error handling to maintain clean and readable code.
- **Winner Check:** Checking for a winner involves iterating over possible sequences of four chips. This can be optimized by limiting the range of iteration based on recent moves, though this basic implementation should suffice for typical gameplay scenarios.

T3 Testing and Verification of Programs via appropriate tools and techniques

Test Case ID	Test Case Description	Test Steps	Test Data	Expected Results
TC1	Vertical Win	1. Two players take turns making valid moves vertically.	Valid moves in a single column	The game detects a vertical connection and declares the corresponding
TC2	Horizontal Win	1. Two players take turns making valid	Valid moves in a single row	The game detects a horizontal connection

		moves horizontally.		and declares the corresponding
TC3	Diagonal Win (Upper Left to Lower Right)	1. Two players take turns making valid moves diagonally (UL to LR).	Valid moves diagonally from UL to LR	The game detects a diagonal connection from the upper left to the lower
TC4	Diagonal Win (Upper Right to Lower Left)	1. Two players take turns making valid moves diagonally (UR to LL).	Valid moves diagonally from UR to LL	The game detects a diagonal connection from the upper right to the lower
TC5	Draw	1. Two players take turns making valid moves until the board is filled.	Alternating valid moves until the board	The game detects a draw condition and declares the game as a draw.
TC6	Invalid Input Handling	1. Enter non-existent column letter.	Invalid column letter	The game prompts the player to enter a valid column letter.